APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/720,506	11/24/2003	Mitica Manu	MSFT-2792/306045	4588

41505 7590 09/29/2009
WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION)
CIRA CENTRE, 12TH FLOOR
2929 ARCH STREET
PHILADELPHIA, PA 19104-2891

EXAMINER	
CHEN, QING	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
09/29/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.**	**Applicant(s)**	
	10/720,506	MANU, MITICA	
	Examiner	**Art Unit**	
	Qing Chen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 June 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,2,4,8,9,12-15 and 17-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,2,4,8,9,12-15 and 17-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the amendment filed on June 2, 2009.

2. **Claims 1, 2, 4, 8, 9, 12-15, and 17-22** are pending.

3. **Claims 1, 2, 4, 8, 9, 12, 20, and 22** have been amended.

4. **Claims 3, 5-7, 10, 11, and 16** have been canceled.

5. The objections to Claims 1, 2, 4, 8, 9, and 11-22 are withdrawn in view of Applicant's amendments to the claims or cancellation of the claims. However, Applicant's amendments to Claim 9 fail to address the objection due to improper antecedent basis. Accordingly, this objection is maintained and further explained hereinafter.

6. The 35 U.S.C. § 112, second paragraph, rejections of Claims 1, 2, 8, 9, and 11-21 are withdrawn in view of Applicant's amendments to the claims or cancellation of the claims. However, Applicant's amendments to Claims 4 and 22 fail to address the rejections due to indefinite claim language. Accordingly, these rejections are maintained and further explained hereinafter.

7. For clarity of the prosecution history record, it is noted that Applicant provides a response to the 35 U.S.C. § 112, first paragraph, rejections of Claims 1, 2, 4, 8, 9, and 11-22 in the "Remarks" (received on 06/02/2009). However, Claims 1, 2, 4, 8, 9, and 11-22 are <u>not</u> rejected under 35 U.S.C. § 112, first paragraph, in the Non-Final Rejection (mailed on 03/02/2009).

Response to Amendment

Claim Objections

8. **Claims 4 and 9** are objected to because of the following informalities:

- **Claim 4** contains a typographical error: "[A] plurality of procedural-oriented programming language" should read -- a plurality of procedural-oriented programming *languages* --.
- **Claim 9** recites the limitation "the processed block of programming code." Applicant is advised to change this limitation to read "the processed block of procedural-oriented programming code" for the purpose of providing it with proper explicit antecedent basis. Appropriate correction is required.

Claim Rejections - 35 USC § 112

9. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

10. **Claims 4 and 22** are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 4 recites the limitation "each of the at least one the at least one of a plurality of procedural-oriented programming languages." This is awkward claim language and thus, renders the claim indefinite. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "the at least one of a plurality of procedural-oriented programming languages" for the purpose of further examination.

Claim 22 recites the limitation "wherein the functional software model comprises a graphical representation of the plurality of code elements and flow of the processed block of procedural-oriented programming code comprising the functional software model." This is awkward and redundant claim language and thus, renders the claim indefinite. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "wherein the functional software model comprises a graphical representation of the plurality of code elements and flow of the processed block of procedural-oriented programming code" for the purpose of further examination.

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. **Claims 1, 2, 4, 8, 9, 12, 15, 17-20, and 22** are rejected under 35 U.S.C. 103(a) as being unpatentable over **US 2004/0031015 (hereinafter "Ben-Romdhane")** in view of **US 6,651,246 (hereinafter "Archambault")** and **US 5,675,801 (hereinafter "Lindsey")**.

As per **Claim 1**, Ben-Romdhane discloses:

- a computer display *(see Paragraph [0207], "Preferably, the information model is presented on the display of the client computer that selected the particular information model.")*; and

- a modeler for defining at least one of a plurality of code elements and a structure of a code block and generating a graphical representation on said computer display of the at least one of a plurality of code elements and structure of the code block, wherein the modeler processes input comprising a code block of procedural-oriented source code and generates from the input a functional model comprising a graphical representation of a structure and flow of the code block *(see Figure 1; Paragraph [0053], "For example, one aspect of the invention disclosed herein allows for the source code of a software program to be broken down into its basic components and graphically presented in a fashion that describes the interdependencies between the components of the software program and the high level procedural flow of the program."; Paragraph [0056], "In this exemplary embodiment, a software application has a set of source code files 1 that comprise the entire application. Source code 1 is analyzed by model generator 2 to create information model 3. Information model 3 can then be presented to a user through model viewer 4."; Paragraph [0120], "Information model viewer 4 provides a graphical presentation of the information model generated by the generator 2. The viewer 4 may present a visual diagram of the software architecture that is inherent in the body of source code. For example, viewer 4 may graphically represent the components derived from the body of source code by generator 2. Additionally, viewer 4 may graphically represent the relationship of each component to the other components in the software architecture."; Paragraph [0143], "For* *example, native source code in a procedural programming language such as COBOL, FORTRAN, Pascal, Java, or C could be presented according to a projected organization of the procedural programming language source code in an object oriented programming paradigm.").*

However, Ben-Romdhane does not disclose:

- wherein the modeler processes input comprising a code block of procedural-oriented source code from an innermost element to an outermost element;

- a selector for selecting at least one of a plurality of procedural-oriented programming languages in which to generate procedural-oriented output source code from the functional model; and

- a code generator for generating procedural-oriented output source code from the functional model.

Archambault discloses:

- processing source code from an innermost element to an outmost element *(see Column 6: 38-41, "As indicated in FIG. 1, the loop allocation of the preferred embodiment creates PDG 14 from nested source code 10. PDG builder 12 starts with the innermost nested loop and moves outwards.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Archambault into the teaching of Ben-Romdhane to modify Ben-Romdhane's invention to include wherein the modeler processes input comprising a code block of procedural-oriented source code from an innermost element to an outermost element. The modification would be obvious because one of ordinary skill in the art would be motivated to process all the code elements of the source code.

Lindsey discloses:

- a selector for selecting at least one of a plurality of procedural-oriented programming languages in which to generate procedural-oriented output source code from a functional model *(see Column 7: 2-5, "One set of source code templates is provided for each target language available through the generator tool 50. Typically, the target language will be a 3GL, such as C or COBOL." and 52-53, "The user also specifies the target language via the user interface 60 (Step 102).")*; and

- a code generator for generating procedural-oriented output source code from the functional model *(see Column 8: 44-49, "When it is finally determined in Step 130 that there are no additional logic objects from the object oriented model that require mapping, the mapped source code templates are parsed by the generator engine 64 in accordance with its parsing algorithm (Step 134) and the resulting source code is output (Step 136).")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Lindsey into the teaching of Ben-Romdhane to modify Ben-Romdhane's invention to include a selector for selecting at least one of a plurality of procedural-oriented programming languages in which to generate procedural-oriented output source code from the functional model; and a code generator for generating procedural-oriented output source code from the functional model. The modification would be obvious because one of ordinary skill in the art would be motivated to utilize a source code generator engine which outputs source code in a specific target language having functionality corresponding to an input design *(see Lindsey – Column 2: 7-12).*

As per **Claim 2**, the rejection of **Claim 1** is incorporated; however, Ben-Romdhane and Archambault do not disclose:

- a user interface for receiving the definition of the at least one of a plurality of code elements and the structure of the code block.

Lindsey discloses:

- a user interface for receiving a definition of at least one of a plurality of code elements and structure of a code block *(see Column 6: 17-21, "In a different implementation of the object oriented user interface 60, the user can be required to directly manipulate the methods or classes of the underlying object oriented language in order to specify the desired programming function.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Lindsey into the teaching of Ben-Romdhane to modify Ben-Romdhane's invention to include a user interface for receiving the definition of the at least one of a plurality of code elements and the structure of the code block. The modification would be obvious because one of ordinary skill in the art would be motivated to utilize a source code generator engine which outputs source code in a specific target language having functionality corresponding to an input design *(see Lindsey – Column 2: 7-12).*

As per **Claim 4**, the rejection of **Claim 1** is incorporated; however, Ben-Romdhane and Archambault do not disclose:

- a code generator for receiving the graphical representation of the at least one of a plurality of code elements and the structure of the code block and the at least one of a plurality of procedural-oriented programming languages and generating procedural-oriented output source code in the at least one of a plurality of procedural-oriented programming languages.

Lindsey discloses:

- a code generator for receiving a graphical representation of at least one of a plurality of code elements and structure of a code block and at least one of a plurality of procedural-oriented programming languages and generating procedural-oriented output source code in the at least one of a plurality of procedural-oriented programming languages *(see Column 8: 44-49, "When it is finally determined in Step 130 that there are no additional logic objects from the object oriented model that require mapping, the mapped source code templates are parsed by the generator engine 64 in accordance with its parsing algorithm (Step 134) and the resulting source code is output (Step 136).")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Lindsey into the teaching of Ben-Romdhane to modify Ben-Romdhane's invention to include a code generator for receiving the graphical representation of the at least one of a plurality of code elements and the structure of the code block and the at least one of a plurality of procedural-oriented programming languages and generating procedural-oriented output source code in the at least one of a plurality of procedural-oriented programming languages. The modification would be obvious because one of ordinary skill in the art would be motivated to utilize a source code generator engine which outputs source code in a specific target language having functionality corresponding to an input design *(see* *Lindsey – Column 2: 7-12).*

As per **Claim 8**, Ben-Romdhane discloses:

- processing a block of procedural-oriented programming code and generating from the processed block of procedural-oriented programming code a functional software model *(see Paragraph [0056], "In this exemplary embodiment, a software application has a set of source code files 1 that comprise the entire application. Source code 1 is analyzed by model generator 2 to create information model 3. Information model 3 can then be presented to a user through model viewer 4."; Paragraph [0143], "For example, native source code in a procedural programming language such as COBOL, FORTRAN, Pascal, Java, or C could be presented according to a projected organization of the procedural programming language source code in an object oriented programming paradigm.")*;
- defining a plurality of code elements within the processed block of procedural-oriented programming code *(see Paragraph [0082], "Once the files in source code 1 have been accessed, generator 2 may extract the control flow, functional dependencies, and data dependencies from the individual files, organize related files or subsets thereof into components, and create the information model.")*;
- specifying a structure of the processed block of procedural-oriented programming code including the plurality of code elements *(see Paragraph [0082], "Once the files in source code 1 have been accessed, generator 2 may extract the control flow, functional dependencies,*

and data dependencies from the individual files, organize related files or subsets thereof into components, and create the information model."); and

- generating from the plurality of code elements and the structure of the processed block of procedural-oriented programming code including the plurality of code elements a functional software model, wherein the functional software model comprises a graphical representation of the plurality of code elements and flow of the processed block of procedural-oriented programming code *(see Paragraph [0056], "In this exemplary embodiment, a software application has a set of source code files 1 that comprise the entire application. Source code 1 is analyzed by model generator 2 to create information model 3. Information model 3 can then be presented to a user through model viewer 4."; Paragraph [0120], "Information model viewer 4 provides a graphical presentation of the information model generated by the generator 2. The viewer 4 may present a visual diagram of the software architecture that is inherent in the body of source code. For example, viewer 4 may graphically represent the components derived from the body of source code by generator 2. Additionally, viewer 4 may graphically represent the relationship of each component to the other components in the software architecture.").*

However, Ben-Romdhane does not disclose:

- processing a block of procedural-oriented programming code from an innermost element to an outermost element;

- specifying at least one procedural-oriented target language in which procedural-oriented output source code for the graphical representation is to be generated, wherein the at least one procedural-oriented target language specified is different from a language of the processed block of procedural-oriented programming code; and

- generating procedural-oriented output source code in the at least one procedural-oriented target language from the functional software model.

Archambault discloses:

- processing source code from an innermost element to an outmost element *(see Column 6: 38-41, "As indicated in FIG. 1, the loop allocation of the preferred embodiment creates PDG 14 from nested source code 10. PDG builder 12 starts with the innermost nested loop and moves outwards.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Archambault into the teaching of Ben-Romdhane to modify Ben-Romdhane's invention to include processing a block of procedural-oriented programming code from an innermost element to an outermost element. The modification would be obvious because one of ordinary skill in the art would be motivated to process all the code elements of the source code.

Lindsey discloses:

- specifying at least one procedural-oriented target language in which procedural-oriented output source code for a graphical representation is to be generated, wherein the at least one procedural-oriented target language specified is different from a language of a processed block of procedural-oriented programming code *(see Column 7: 2-5, "One set of source code templates is provided for each target language available through the generator tool 50. Typically, the target language will be a 3GL, such as C or COBOL." and 52-53, "The user also specifies the target language via the user interface 60 (Step 102).")*; and

- generating procedural-oriented output source code in the at least one procedural-oriented target language from a functional software model *(see Column 8: 44-49, "When it is finally determined in Step 130 that there are no additional logic objects from the object oriented model that require mapping, the mapped source code templates are parsed by the generator engine 64 in accordance with its parsing algorithm (Step 134) and the resulting source code is output (Step 136).").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Lindsey into the teaching of Ben-Romdhane to modify Ben-Romdhane's invention to include specifying at least one procedural-oriented target language in which procedural-oriented output source code for the graphical representation is to be generated, wherein the at least one procedural-oriented target language specified is different from a language of the processed block of procedural-oriented programming code; and generating procedural-oriented output source code in the at least one procedural-oriented target language from the functional software model. The modification would be obvious because one of ordinary skill in the art would be motivated to utilize a source code generator engine which outputs source code in a specific target language having functionality corresponding to an input design *(see Lindsey – Column 2: 7-12).*

As per **Claim 9**, the rejection of **Claim 8** is incorporated; however, Ben-Romdhane and Archambault do not disclose:

- receiving the definition of the plurality of code elements within the processed block of procedural-oriented programming code and specifying the structure of the processed block of procedural-oriented programming code via a user interface.

Lindsey discloses:

- receiving a definition of a plurality of code elements within a processed block of procedural-oriented programming code and specifying a structure of the processed block of procedural-oriented programming code via a user interface *(see Column 6: 17-21, "In a different implementation of the object oriented user interface 60, the user can be required to directly manipulate the methods or classes of the underlying object oriented language in order to specify the desired programming function.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Lindsey into the teaching of Ben-Romdhane to modify Ben-Romdhane's invention to include receiving the definition of the plurality of code elements within the processed block of procedural-oriented programming code and specifying the structure of the processed block of procedural-oriented programming code via a user interface. The modification would be obvious because one of ordinary skill in the art would be motivated to utilize a source code generator engine which outputs source code in a specific target language having functionality corresponding to an input design *(see Lindsey – Column 2: 7-12)*.

As per **Claim 12**, the rejection of **Claim 8** is incorporated; and Ben-Romdhane further discloses:

- wherein one of the plurality of code elements comprises a variable, comment, constant, function, method, member, data type, delegate, reference, field, variant, property, interface, type, enumeration, structure, primitive, array, or event handle *(see Paragraph [0088], "For example, comments from within the source code can be extracted by parser 52 for inclusion in the resulting LDF file.").*

As per **Claim 15**, the rejection of **Claim 8** is incorporated; and Ben-Romdhane further discloses:

- wherein one of the plurality of code elements comprises an evaluation entity *(see Paragraph [0085], "Moreover, parser 52 may determine the names of each of the functions contained within the file and the function calls made by the file.").*

As per **Claim 17**, the rejection of **Claim 8** is incorporated; and Ben-Romdhane further discloses:

- wherein one of the plurality of code elements comprises a passive entity *(see Paragraph [0088], "For example, comments from within the source code can be extracted by parser 52 for inclusion in the resulting LDF file.").*

As per **Claim 18**, the rejection of **Claim 17** is incorporated; and Ben-Romdhane further discloses:

- wherein the passive entity comprises a comment or a modeling diagram *(see Paragraph [0088], "For example, comments from within the source code can be extracted by parser 52 for inclusion in the resulting LDF file.").*

As per **Claim 19**, the rejection of **Claim 8** is incorporated; and <u>Ben-Romdhane</u> further discloses:

- wherein one of the plurality of code elements comprises a block entity *(see Paragraph [0232], "When the parser analyzes a module, it advantageously recognizes explicitly defined and implicitly defined functions within the module. An explicitly defined function can be a procedural function definition in the module (as in a C procedural language module) or an object/class member function definition (as in a C++ object oriented language module).").*

As per **Claim 20**, the rejection of **Claim 19** is incorporated; and <u>Ben-Romdhane</u> further discloses:

- wherein the block entity comprises a method entity, a member entity, or a file entity *(see Paragraph [0232], "When the parser analyzes a module, it advantageously recognizes explicitly defined and implicitly defined functions within the module. An explicitly defined function can be a procedural function definition [a method entity] in the module (as in a C procedural language module) or an object/class member function definition (as in a C++ object oriented language module).").*

As per **Claim 22**, <u>Ben-Romdhane</u> discloses:

- processing a block of procedural-oriented programming code and generating from the processed block of procedural-oriented programming code a functional software model *(see Paragraph [0056], "In this exemplary embodiment, a software application has a set of source code files 1 that comprise the entire application. Source code 1 is analyzed by model generator 2 to create information model 3. Information model 3 can then be presented to a user through model viewer 4."; Paragraph [0143], "For example, native source code in a procedural programming language such as COBOL, FORTRAN, Pascal, Java, or C could be presented according to a projected organization of the procedural programming language source code in an object oriented programming paradigm.")*;

- defining a plurality of code elements within the processed block of procedural-oriented programming code *(see Paragraph [0082], "Once the files in source code 1 have been accessed, generator 2 may extract the control flow, functional dependencies, and data dependencies from the individual files, organize related files or subsets thereof into components, and create the information model.")*;

- specifying a structure of the processed block of procedural-oriented programming code including the plurality of code elements *(see Paragraph [0082], "Once the files in source code 1 have been accessed, generator 2 may extract the control flow, functional dependencies, and data dependencies from the individual files, organize related files or subsets thereof into components, and create the information model.")*; and

- generating from the plurality of code elements and the structure of the processed block of procedural-oriented programming code including the plurality of code elements a functional software model, wherein the functional software model comprises a graphical representation of the plurality of code elements and flow of the processed block of procedural-oriented programming code *(see Paragraph [0056], "In this exemplary embodiment, a software application has a set of source code files 1 that comprise the entire application. Source code 1 is analyzed by model generator 2 to create information model 3. Information model 3 can then be presented to a user through model viewer 4."; Paragraph [0120], "Information model viewer 4 provides a graphical presentation of the information model generated by the generator 2. The viewer 4 may present a visual diagram of the software architecture that is inherent in the body of source code. For example, viewer 4 may graphically represent the components derived from the body of source code by generator 2. Additionally, viewer 4 may graphically represent the relationship of each component to the other components in the software architecture.").*

However, Ben-Romdhane does not disclose:

- processing a block of procedural-oriented programming code from an innermost program element to an outermost program element; and

- generating procedural-oriented output source code from the functional software model.

Archambault discloses:

- processing source code from an innermost element to an outmost element *(see Column 6: 38-41, "As indicated in FIG. 1, the loop allocation of the preferred embodiment creates PDG 14 from nested source code 10. PDG builder 12 starts with the innermost nested loop and moves outwards.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Archambault into the teaching of Ben-

Romdhane to modify Ben-Romdhane's invention to include processing a block of procedural-oriented programming code from an innermost program element to an outermost program element. The modification would be obvious because one of ordinary skill in the art would be motivated to process all the code elements of the source code.

Lindsey discloses:

- generating procedural-oriented output source code from a functional software model *(see Column 8: 44-49, "When it is finally determined in Step 130 that there are no additional logic objects from the object oriented model that require mapping, the mapped source code templates are parsed by the generator engine 64 in accordance with its parsing algorithm (Step 134) and the resulting source code is output (Step 136).")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Lindsey into the teaching of Ben-Romdhane to modify Ben-Romdhane's invention to include generating procedural-oriented output source code from the functional software model. The modification would be obvious because one of ordinary skill in the art would be motivated to utilize a source code generator engine which outputs source code in a specific target language having functionality corresponding to an input design *(see Lindsey – Column 2: 7-12)*.

13. **Claims 13 and 14** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Ben-Romdhane** in view of **Archambault** and **Lindsey** as applied to Claim 8 above, and further in view of **US 6,684,385 (hereinafter "Bailey")**.

As per **Claim 13**, the rejection of **Claim 8** is incorporated; however, Ben-Romdhane, Archambault, and Lindsey do not disclose:

- wherein one of the plurality of code elements comprises a code relation.

Bailey discloses:

- wherein one of a plurality of code elements comprises a code relation *(see Column 8: 30-36, "... each program object typically performs some useful function, such as a Boolean operation (e.g., AND, OR, etc.), a mathematical operation, a data acquisition operation ..., renders some comparison (e.g., less than, greater than, equal to, etc.), and so on.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bailey into the teaching of Ben-Romdhane to modify Ben-Romdhane's invention to include wherein one of the plurality of code elements comprises a code relation. The modification would be obvious because one of ordinary skill in the art would be motivated to model all aspects of a software program.

As per **Claim 14**, the rejection of **Claim 13** is incorporated; however, Ben-Romdhane, Archambault, and Lindsey do not disclose:

- wherein the code relation comprises a mathematical operator.

Bailey discloses:

- wherein a code relation comprises a mathematical operator *(see Column 8: 30-36, "... each program object typically performs some useful function, such as a Boolean operation (e.g., AND, OR, etc.), a mathematical operation, a data acquisition operation ..., renders some comparison (e.g., less than, greater than, equal to, etc.), and so on.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bailey into the teaching of Ben-Romdhane to modify Ben-Romdhane's invention to include wherein the code relation comprises a mathematical operator. The modification would be obvious because one of ordinary skill in the art would be motivated to model all aspects of a software program.

14. **Claim 21** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Ben-Romdhane** in view of **Archambault** and **Lindsey** as applied to Claim 20 above, and further in view of **US 6,199,195 (hereinafter "Goodwin")**.

As per **Claim 21**, the rejection of **Claim 20** is incorporated; however, Ben-Romdhane, Archambault, and Lindsey do not disclose:

- wherein a many-to-many relationship exists between block entities.

Goodwin discloses:

- wherein a many-to-many relationship exists between block entities *(see Column 4: 31-36, "A 'relationship' defines a link between two object classes." and "Relationships can be one-to-one, one-to-many, or many-to-many.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Ben-Romdhane to modify Ben-Romdhane's invention to include wherein a many-to-many relationship exists between block entities. The modification would be obvious because one of ordinary skill in the art would be motivated to link classes to other classes.

Response to Arguments

15. Applicant's arguments filed on June 2, 2009 have been fully considered, but they are not persuasive.

In the Remarks, Applicant argues:

a) Thus, Applicants submit that Ben-Romdhane does not teach a modeler for defining a plurality of code elements nor does Ben-Romdhane teach specifying a structure of a code block that includes those code elements. As such, Applicants submit that Ben-Romdhane does not teach a graphical representation of the code elements and the structure of the code block because Ben-Romdhane never reaches that level of detail in its organized view of source files. Rather, Ben-Romdhane is limited to the organization of the individual source files, not the organization of the code elements within the source files.

Examiner's response:

a) Examiner disagrees. With respect to the Applicant's assertion that Ben-Romdhane does not teach a graphical representation of the code elements and the structure of the code block, as previously pointed out in the Non-Final Rejection (mailed on 03/02/2009) and further clarified hereinafter, the Examiner respectfully submits that Ben-Romdhane clearly discloses a graphical representation of the code elements and the structure of the code block *(see Figure 1; Paragraph [0053], "For example, one aspect of the invention disclosed herein allows for the source code of a software program to be broken down into its basic components and graphically presented in a* *fashion that describes the interdependencies between the components of the software program and the high level procedural flow of the program."; Paragraph [0056], "In this exemplary embodiment, a software application has a set of source code files 1 that comprise the entire application. Source code 1 is analyzed by model generator 2 to create information model 3. Information model 3 can then be presented to a user through model viewer 4."; Paragraph [0120], "Information model viewer 4 provides a graphical presentation of the information model generated by the generator 2. The viewer 4 may present a visual diagram of the software architecture that is inherent in the body of source code. For example, viewer 4 may graphically represent the components derived from the body of source code by generator 2. Additionally, viewer 4 may graphically represent the relationship of each component to the other components in the software architecture.").* Attention is drawn to Figure 1 of Ben-Romdhane which clearly illustrates that source code is analyzed by a model generator to create an information model (graphical representation). Thus, one of ordinary skill in the art would readily comprehend that the source code would contain various code elements and structure of code blocks. Furthermore, note that the information model is derived from the software architecture that is inherent in the body of the source code. Thereby, the source code of a software program can be graphically presented in a fashion that describes the interdependencies between the components of the software program and the high-level procedural flow of the software program.

Therefore, for at least the reason set forth above, the rejections made under 35 U.S.C. § 103(a) with respect to Claims 1, 8, and 22 are proper and therefore, maintained.

Conclusion

16. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

17. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Q. C./

Examiner, Art Unit 2191

/Anna Deng/

Primary Examiner, Art Unit 2191